ROBOFUN.RO

LECȚIA IV

CURS GRATUIT

ARDUINO ȘI ROBOTICĂ

Introducere în Arduino

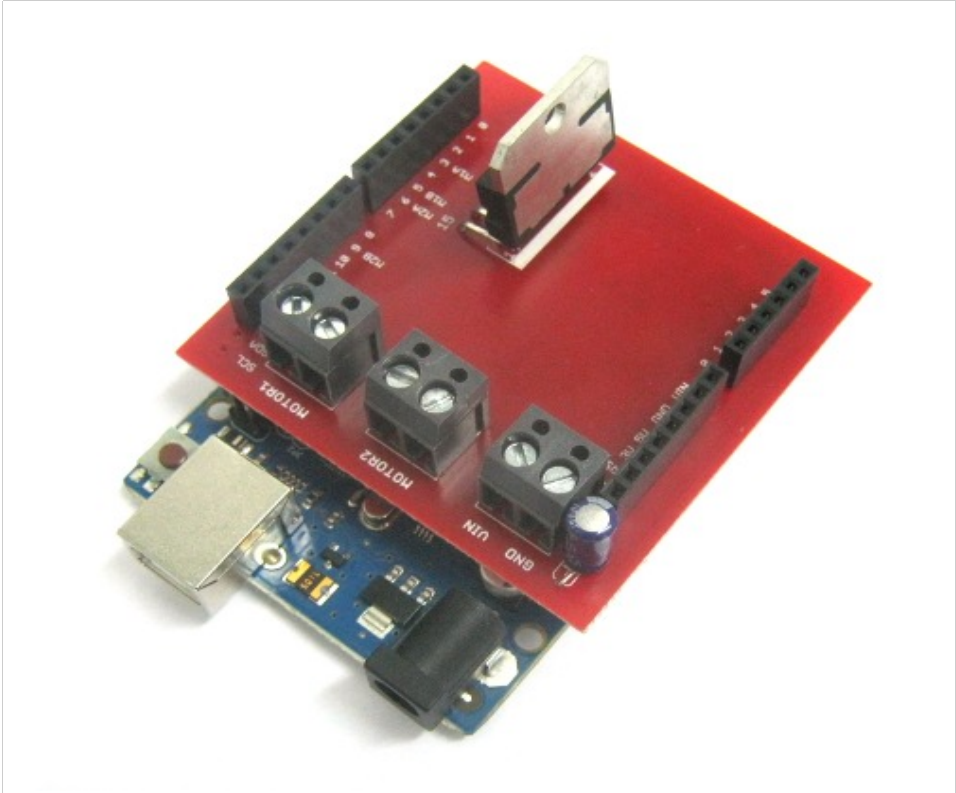
Modalități de comunicare între Arduino și componentele din ecosistem

Despre Shield-uri Arduino

Shield-urile Arduino reprezinta o modalitate foarte simpla de a extinde functionalitatile Arduino. Orice shield Arduino este standard, in sensul ca pinii acestuia se potrivesc perfect peste pinii placii Arduino. In plus, in marea majoritate a cazurilor, pinii shield-ului sunt prelungiti in sus cu conectori de tip mama. Acesti pini, fiind infipti in pinii Arduino, ii prelungesc practic in sus pe acestia. Astfel, daca am pus un shield peste Arduino si vrem sa conectam un senzor la pinul A0 al Arduino, atunci vom conecta senzorul la pinul A0 al shield-ului, si totul va functiona exact ca si cum l-am fi infipt in pinul A0 al Arduino.

Fiecare shield utilizeaza anumiti pini pentru a se conecta cu Arduino. Astfel, spre exemplu, shield-ul cu LCD 16X2 utilizeaza pinii 7, 6, 5, 4, 3, 2 iar shield-ul Wifly foloseste pinii 10, 11, 12, 13. Acesti pini nu mai pot fi folositi in acelasi timp cu shield-ul. Astfel, daca avem un senzor care se conecteaza pe pinul 3 la Arduino, si noi folosim deja shield-ul cu L298 (care foloseste pinii 3, 5, 6 si 9), atunci nu vom putea conecta senzorul in acelasi timp cu shield-ul L298. Mai mult, se poate intampla chiar sa avem doua shield-uri care utilizeaza aceeasi pini (sau o parte dintre pini sunt aceeasi). In aceasta situatie, de cele mai multe ori se poate aplica o solutie clasica. Anume, pinii shield-ului care intra in Arduino se indoaie un pic in afara, astfel incat sa nu mai intre in Arduino, ci sa iasa in afara. Apoi, acesti pini se conecteaza pe deasupra cu pini tata-tata cu alti pini Arduino nefolositi. De la caz la caz, s-ar putea sa fie nevoie sa faci modificari in libraria asociata senzorului, sau sa modifici doar codul sursa din programul tau, astfel incat pinii pe care i-ai mutat in alta parte sa fie actualizati si in cod.

Mai jos sunt doua exemple de shield-uri montate peste Arduino.



Modalitati de Comunicare Intre Arduino si Componente

Mai departe vom trece in revista cateva modalitati prin care Arduino se conecteaza cu celelalte componente (senzori, shield-uri, etc). S-ar putea ca atunci cand vom ajunge la comunicarea seriala, SPI si I2C lucrurile sa ti se para un pic complicate si dificil de inteles, pentru ca inca nu am introdus nici un exemplu de dispozitiv care sa comunice astfel. Dupa ce vei vedea mai multe dispozitive explicate in detaliu in lectiile viitoare, vei intelege mai usor despre ce este vorba.

Citire Date pe Porturile Analogice sau Digitale

Anumite componente se conecteaza direct la porturile digitale sau analogice ale Arduino si astfel primesc comenzi sau comunica date catre acesta. Astfel, un led se conecteaza la un pin digital si atunci cand pinul este in HIGH, led-ul se aprinde. Un senzor de lumina se conecteaza la un pin analogic si folosind "analogRead(<pin>)" obtii valoarea data de senzor. Un buton se conecteaza la un pin digital si "digitalRead(<pin>)" intoarce 1 daca butonul este apasat si 0 altfel. Un shield driver de motoare cu L298 sau un LCD se conecteaza la mai multi pini digitali (4, respectiv 7) si primeste comenzi de la

Arduino prin intermediul acestora.

Voi reaminti ca ori de cate ori avem nevoie sa folosim un pin digital ca sa comandam un dispozitiv, este necesar sa declaram acest pin ca fiind de tip OUTPUT ("pinMode(<pin>, OUTPUT"). Altfel, nu se va intampla absolut nimic atunci cand trecem pinul in HIGH sau in LOW.

Merita sa insistam un pic si asupra citirii datelor pe porturile analogice. Un senzor, la nivel electric, da pe iesire o tensiune electrica. Este absolut necesar ca aceasta tensiune sa fie intre 0 si 5 V. Orice senzor special creat pentru Arduino va respecta deja aceasta regula. Mai departe, aceasta tensiune este preluata de o componenta specializata a lui Arduino, numita "convertorul analog-digital". Aceasta componenta ofera la iesirea sa un numar intre 0 si 1023, in functie de nivelul de tensiune de la intrare. Asftel, folosind Arduino, putem citi senzori analogici cu o precizie de exact 1024 de nivele distincte. De exemplu, folosind senzorul de lumina, vom putea citi cu Arduino exact 1024 de nivele distincte de iluminare ale mediului ambiant. Daca avem nevoie de o precizie mai buna, atunci va trebui sa trecem la alt tip de senzori (cel mai probabil, senzori pe I2C).

Comunicare Seriala

Comunicarea seriala este o comunicare de tip digital, in sensul ca pe fir se transmit biti (ca la fel cum se intampla intr-o retea de calculatoare, doar ca folosind un protocol mult simplificat). Aceasta comunicare seriala se intampla intotdeauna intre Arduino si calculator, atunci cand Arduino este programat. O comunicare seriala se intampla intotdeauna la o anumita rata de transfer, care determina viteza cu care se transmit bitii pe fir. Rate comune de transfer sunt 9600, 19200, 57600, 115200.

Software Serial

SoftwareSerial este tot o modalitate de comunicare seriala, doar ca in loc de a utiliza pinii RX si TX ai microcontroller-ului (care sunt dedicati pentru o astfel de comunicare), utilizam o librarie software care emuleaza comunicarea folosind (aproape) oricare doi pini digitali ai placii Arduino. Avantajul imediat este faptul ca in acest mod putem conecta placa Arduino cu mult mai multe dispozitive seriale.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
void setup()  {
  Serial.begin(9600);
  Serial.println("Goodnight moon!");
  mySerial.begin(4800);
  mySerial.println("Hello, world?");
}

void loop() {
    if (mySerial.available()) {
        Serial.write(mySerial.read());
    }
    if (Serial.available()) {
        mySerial.write(Serial.read());
    }
}
```

In codul de mai sus am declarat pinii 10 si 11 ca fiind cei doi pini care vor emula comunicarea seriala. Pinul 10 este pinul RX iar pinul 11 este pinul TX. Mai departe, am declarat rata de transfer a acestei conexiuni seriale ca fiind 4800 (*mySerial.begin(4800)*) (rata de transfer o stim din specificatiile dispozitivului cu care urmeaza sa ne conectam). In continuare, orice caracter primit peste conexiunea seriala emulata este scrisa in Serial Monitor, si orice caracter introdus in Serial Monitor este trimis peste conexiunea seriala emulata software.

Vei intalni des in proiectele de mai departe acest tip de conexiune. Chiar daca in proiectul respectiv este folosita conexiunea de tip serial standard, vei putea oricand sa o inlocuiesti cu SoftwareSerial, ca mai sus.

SPI

SPI este o modalitate de a comunica cu dispozitive de tipul master-slave. Un dispozitiv este master (de obicei Arduino), si celelalte dispozitive sunt slave. In practica, vei gasi de obicei librarii scrise de altcineva care incapsuleaza deja comunicare cu dispozitivul peste SPI, asa ca nu va trebui sa iti bati prea mult capul cu amanuntele. Voi prezenta totusi mai jos cateva elemente ale SPI, ca sa recunosti termenii atunci cand auzi vorbindu-se de ei.

Comunicarea SPI se face folosind patru canale (fire de conexiune) :

MISO – firul prin care dispozitivele slave trimit date catre master.

MOSI – firul prin care dispozitivul master trimite date catre slave.

SCK – firul prin catre este transmis clock-ul (ritmul de comunicatie)

SS (Slave Select) – acest canal este specific SPI, si este interesant. SPI permite conectarea pe acelasi bus a mai multor dispozitive, fiecare dispozitiv avand atasat un canal Slave Select. Atunci cand valoarea din acest canal este LOW, dispozitivul slave comunica cu master-ul. Cand valoarea Slave Select este HIGH, atunci dispozitivul ignora comunicarea SPI. In acest mod, poti conecta pe acelasi canal SPI oricat de multe dispozitive, cu conditia ca fiecarui dispozitiv sa ii atasezi un canal Slave Select. Un

exemplu pe care il vei intalni este comunicarea cu shield-ul Ethernet, si in acelasi timp cu SD card-ul montat pe shield. Ambele folosesc comunicare SPI. Pinul Slave Select conectat la chip-ul Ethernet este pinul digital 10 Arduino, iar pinul Slave Select conectat la SD card este pinul digital 4. Astfel, atunci cand vrei sa comunici Ethernet, vei seta pinul 10 in LOW, si pinul 4 in HIGH (doar chip-ul Ethernet activ), iar atunci cand vrei sa comunici cu SD card-ul, vei seta pinul 10 in HIGH si pinul 4 in LOW (doar SD card-ul activ). Daca omiti sa faci acest lucru, comenzile trimise catre chip-ul Ethernet vor fi in acelasi timp receptionate si procesate si de catre SD card, ceea ce va conduce la o functionare defectuoasa.

In cazul Arduino UNO, pinii Arduino SPI sunt MISO – pinul digital 12, MOSI – pinul digital 11, SCK – pinul digital 13. Pinul default SS este pinul digital 10. In cazul in care ai mai mult de un dispozitiv SPI, vei folosi mai multi pini Slave Select, nu doar pinul digital 10, fiecare dispozitiv fiind conectat cu cate un pin Slave Select distinct.

In cazul Arduino Mega, pinii SPI sunt MISO - pinul digital 50, MOSI – pinul digital 51, SCK – pinul digital 52, si SS – pinul digital 53.

La modul concret, daca ai doua dispozitive SPI, vei incepe prin a identifica pinii MOSI, MISO, SCK si SS pentru fiecare dispozitiv. Vei conecta apoi impreuna pinii MOSI ai celor doua dispozitive, si impreuna cu pinul Arduino MOSI, la fel si pentru MISO si SCK. Pentru pinii SS, vei conecta pinul primului dispozitiv la un pin digital Arduino, iar pinul celui de-al doilea dispozitiv la un alt pin digital Arduino.

I2C

Comunicarea I2C este un mod interesant de comunicare, in sensul ca necesita doar doua fire de comunicare, si permite conectarea a oricat de mult dispozitive (cu adrese diferite). Fiecare dispozitiv I2C are o adresa (care se poate sau nu modifica). De obicei, vei intalni dispozitive I2C cu adresa fixa (pe care nu o poti modifica), sau dispozitive la care vei putea alege adresa I2C dintr-o lista de cateva adrese prestabilite (lucru pe care de obicei il faci prin conectarea unui pin de adresa – marcat de obicei ADDR – la pinul GND, sau VCC, sau SDA, sau SCL).

Ca si in cazul SPI, a intelege modul de comunicare citind datasheet-ul dispozitivului I2C nu este facil. Primul lucru pe care il fac atunci cand am nevoie sa comunic cu un dispozitiv I2C este sa caut o librarie deja scrisa pentru dispozitivul respectiv. In acest mod, singurul lucru de care trebuie eu sa tin cont este adresa dispozitivului. Mai departe, modul concret de comunicare este realizat in intregime de librarie, eu avand acces doar la informatia utila.

In cazul Arduino UNO, pinii I2C pentru Arduino sunt pinul analogic 4 (care este pinul SDA), si pinul analogic 5 (care este pinul SCL). Pentru Arduino Mega, pinul digital 20 este pinul SDA si pinul digital 21 este pinul SCL. In cazul Arduino Leonardo, cei doi pini SDA si SCL sunt distincti, marcati ca atare pe placa. Pentru a folosi un dispozitiv I2C, vei conecta pinul SDA al dispozitivului cu pinul SDA al Arduino, pinul SCL al dispozitivului cu pinul SCL al Arduino. In plus, in functie de dispozitiv, s-ar putea sa fie necesar sa conectezi si pinul de adresa la GND sau la VCC (in cele mai multe cazuri, pinul de adresa este deja conectat, dar nu intotdeauna).

Un exemplu de senzor I2C este senzorul de temperatura I2C. Pinul de adresa este marcat pe placa cu “ADDR0”. Dispozitivul acest pin se poate conecta la pinul GND (si atunci dispozitivul va avea

adresa “0x48”), poate fi conectat la VCC (si atunci dispozitivul va avea adresa “0x49”, poate fi conectat la pinul SDA (si atunci dispozitivul va avea adresa “0x4A”), sau poate fi conectat la pinul SCL (si atunci dispozitivul va avea adresa “0x4C”). In acest mod, poti conecta pana la patru senzori TMP102 la placa Arduino.

Daca esti curios, un exemplu de cod care comunica pe I2C poti gasi aici :

http://www.robofun.ro/senzor_temperatura_tmp102

Aceasta a fost lectia 4. In final, as vrea sa te rog sa ne oferi feedback asupra acestei lectii, pentru a ne permite sa le facem mai bune pe urmatoarele.

Este vorba despre un sondaj cu 4 intrebari (oricare este optionala), pe care il poti accesa dand click aici.

Sau ne poti contacta direct prin email la contact@robofun.ro .

Iti multumim,

Echipa Robofun.RO